Class 6: R Functions

Pierce Ford (PID: A59010464)

10/15/2021

Quick Rmarkdown Intro

We can write text of course just like any file. We can **style text to bold** or *italic*.

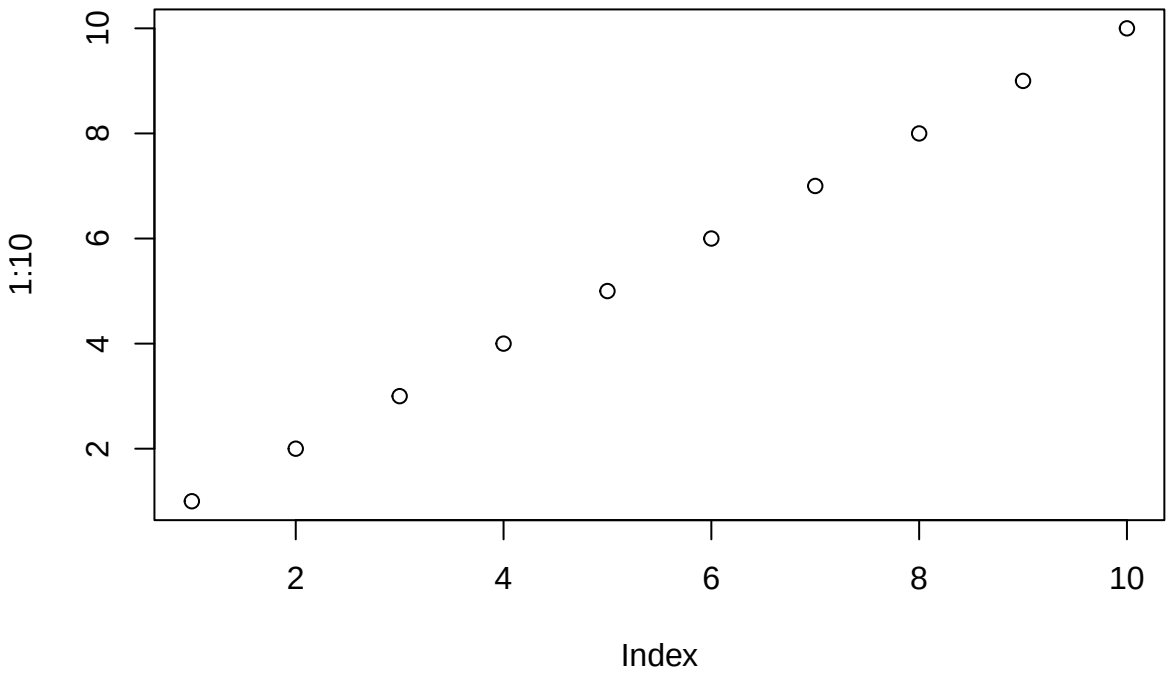
Do:

- this
- and that
- and another thing

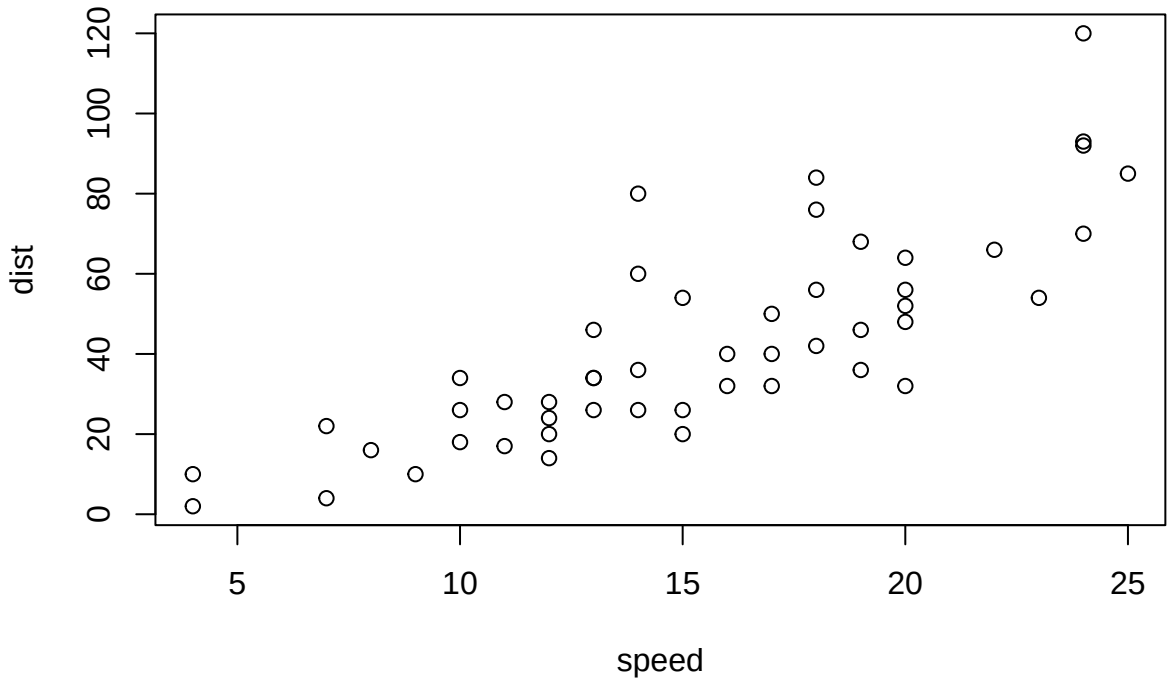
This is more text
and this is a new line

We can include some code:

```
plot(1:10)
```




```
plot(cars)
```



Time to write a function

> Q1. Write a function grade() to determine an overall grade from a vector of student homework assignment scores dropping the lowest single score. If a student misses a homework (i.e. has an NA value) this can be used as a score to be potentially dropped. Your final function should be adquately explained with code comments and be able to work on an example class gradebook such as this one in CSV format: "https://tinyurl.com/gradeinput" [3pts]

```
# Example input vectors to start with
student1 <- c(100, 100, 100, 100, 100, 100, 100, 90)
student2 <- c(100, NA, 90, 90, 90, 90, 97, 80)
student3 <- c(90, NA, NA, NA, NA, NA, NA, NA)
```

First, I want to find the lowest score. I can use **min()** to find it and **which.min()** to find its position.

```
#Calculate and locate min
min(student1)
```

```
## [1] 90
```

```
which.min(student1)
```

```
## [1] 8
```

```
#Doesn't work well when NAs are present
min(student2)
```

```
## [1] NA
```

```
which.min(student2)
```

```
## [1] 8
```

```
min(student3)
```

```
## [1] NA
```

```
which.min(student3)
```

```
## [1] 1
```

Can use [-position] to remove an item (e.g. lowest score).

```
#Example
student1[-which.min(student1)]
```

```
## [1] 100 100 100 100 100 100 100
```

```
#Again, not great with NAs
student2[-which.min(student2)]
```

```
## [1] 100  NA  90  90  90  90  97
```

```
student3[-which.min(student3)]
```

```
## [1] NA NA NA NA NA NA NA
```

Can finish the task for student1, but the NAs cause problems.

```
#Calculate adjusted average
mean(student1[-which.min(student1)])
```

```
## [1] 100
```

```
#Fails completely with NAs
mean(student2[-which.min(student2)])
```

```
## [1] NA
```

```
mean(student3[-which.min(student3)])
```

```
## [1] NA
```

Attempt to set NA values to 0.

```
#What does is.na() return?
is.na(student2)
```

```
## [1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

```
#Returns a logical vector with TRUE where NA elements are present

#Attempt to set
student2_copy <- student2
student2_copy[is.na(student2_copy)] = 0
student2_copy
```

```
## [1] 100   0  90  90  90  90  97  80
```

Apply full method to all three students.

```
#Copy students to preserve originals
student1_copy <- student1
student2_copy <- student2
student3_copy <- student3

#Set NAs in the copies to 0
student1_copy[is.na(student1_copy)] = 0
student2_copy[is.na(student2_copy)] = 0
student3_copy[is.na(student3_copy)] = 0

#Remove minimum and calculate adjusted average
mean(student1_copy[-which.min(student1_copy)])
```

```
## [1] 100
```

```
mean(student2_copy[-which.min(student2_copy)])
```

```
## [1] 91
```

```
mean(student3_copy[-which.min(student3_copy)])
```

```
## [1] 12.85714
```

Dealing with improper inputs.

```
#Convert string to integer
student4 <- c(100, NA, 90, "90", 90, 90, 97, 80)
student4_numeric <- as.numeric(student4)
student4_numeric
```

```
## [1] 100  NA  90  90  90  90  97  80
```

Now turn this process into a function.

```
#Name the function
grade <- function(student){
  #Copy to preserve original
  temp <- student
  #Convert accidental string inputs to values
  temp <- as.numeric(temp)
  #Replace NA with 0
  temp[is.na(temp)]=0
  #Remove minimum and return adjusted average grade
  mean(temp[-which.min(temp)])
}

#Test function
grade(student1)
```

```
## [1] 100
```

```
grade(student2)
```

```
## [1] 91
```

```
grade(student3)
```

```
## [1] 12.85714
```

```
grade(student4)
```

```
## [1] 91
```

Q2. Using your grade() function and the supplied gradebook, Who is the top scoring student overall in the gradebook? [3pts]

Test on gradebook using the apply function and print out top scoring student.

```
#Read in data
gradebook <- read.csv("https://tinyurl.com/gradeinput", row.names=1)
#Calculate final grades and print
final_grades <- apply(gradebook, 1, grade)
final_grades
```

```
##  student-1  student-2  student-3  student-4  student-5  student-6  student-7
##      91.75      82.50      84.25      84.25      88.25      89.00      94.00
##  student-8  student-9 student-10 student-11 student-12 student-13 student-14
##      93.75      87.75      79.00      86.00      91.75      92.25      87.75
## student-15 student-16 student-17 student-18 student-19 student-20
##      78.75      89.50      88.00      94.50      82.75      82.75
```

```
#Who scored the best
names(which.max(final_grades))
```

```
## [1] "student-18"
```

Q3. From your analysis of the gradebook, which homework was toughest on students (i.e. obtained the lowest scores overall? [2pts]

```
#Calculate sums of each column
HW_totals <- apply(gradebook, 2, sum, na.rm=TRUE)
#Which HW was hardest
names(which.min(HW_totals))
```

```
## [1] "hw2"
```

Q5. Make sure you save your Rmarkdown document and can click the "Knit" button to generate a PDF foramt report without errors. Finally, submit your PDF to gradescope. [1pt]

All done!!!